The Filter Wizard
issue 7: An Excelent fit, Sir!
Kendall Castor-Perry

The cat is out of the bag. **PSoC3**, Cypress's new family of programmable mixed-signal systems-on-chip, has been announced. Visit our website to learn why, whatever you do in electronics, it'll be a paradigm-buster for you. Phew, it's hot in here, off with that marketing hat...

This is the first of two columns showing a simple technique for designing, optimizing and implementing a response-equalizing filter. The second part will look at implementing it on PSoC3; this part is about the design process. Figure 1 is the first piece of our story. It's the curve of the sensitivity of a sensor versus frequency (what *kind* of sensor, do you think?):

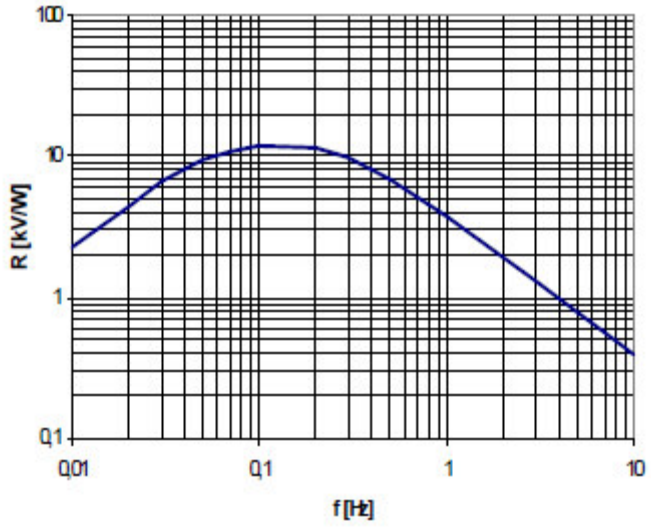


Figure 1: frequency response of the sensor

The customer wanted to equalize this so that the final sensitivity would be "independent of frequency". Now, an elementary rule of Filter Wizardry is: *the frequency response of a filter doesn't stop at the edges of the graph*. It's important to understand what the response should do outside this range. "Don't Care" *doesn't* mean "Don't Need To Care".

Now, I'll wager that the response of that sensor continues to fall off at 20dB/decade outside the graph limits. But it would be a *terrible* idea to *boost* the response with gain that continues to rise without limits. We'll want the response to flatten off or even fall back down, to avoid noise problems. For a first try, let's hold constant the response outside figure 1's frequency range, for one extra decade in each direction; figure 2:

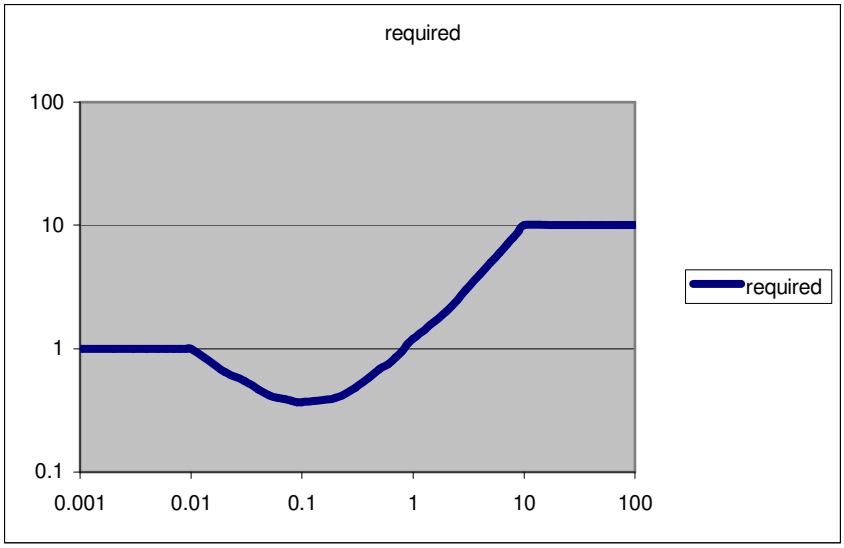


Figure 2: extend the response on either side.

We want to design a realizable filter with a frequency response approximating this curve to an acceptable error. The low frequencies involved indicate using a digital filter to avoid bulky external passive components. That's handy, because PSoC3 has a built-in 24bit digital filtering engine, so fast that it'll hardly have to wake up to crunch out anything I'll ask it to do here. The digital stuff is for the next column; first, let's design it in the analogue domain.

Figure 2's response is flat at one end, wavy in the middle and flat at the other end (apologies to Monty Python); that's a second order 'shelf' equalizer. The amplitude response of the general second order transfer function or 'biquad' section:

$$H(s) = \frac{a2 \cdot s^2 + a1 \cdot s + a0}{b2 \cdot s^2 + b1 \cdot s + b0} \qquad [1]$$

is easily computed in a spreadsheet. With ω equal to the input frequency divided by the frequency to which the transfer function is normalized, a bit of manipulation gives:

$$G(\omega) = \left|H(j\omega)\right| = \sqrt{\frac{(a0 - a2 \cdot \omega^2)^2 + a1^2 \cdot \omega^2}{(b0 - b2 \cdot \omega^2)^2 + b1^2 \cdot \omega^2}} \qquad [2]$$

Figure 3 is a clippet of a working spreadsheet, using one column for each biquad section, with the cell formula shown in glorious Excelcolor. The gains of all the sections used are multiplied up (linear gains, not dB) and deposited in column G:

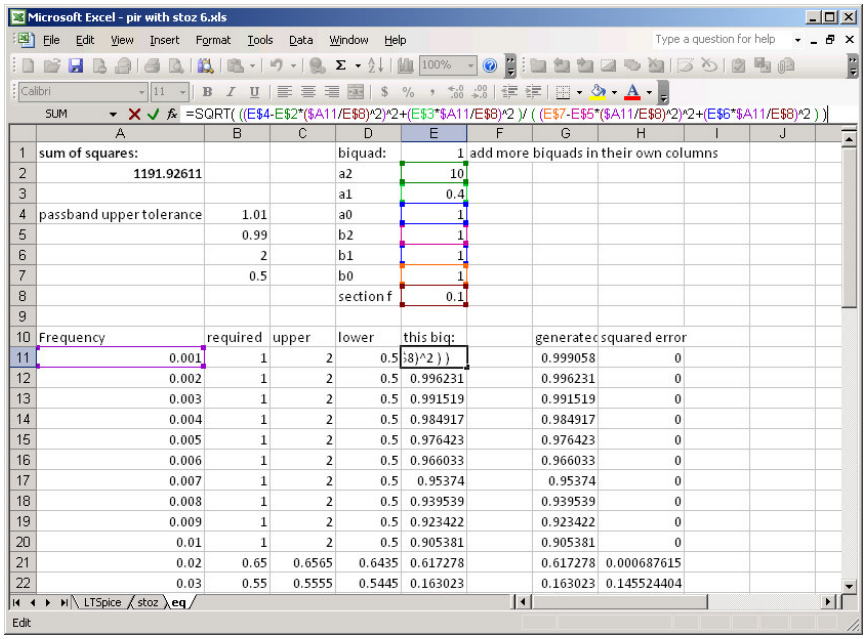


Figure 3: working out the filter response in Excel

Now we need an initial coefficient guess; here's how I'd do it:

1. the middle of the frequency range is around 0.3Hz so that's where we'll normalize everything to.
2. at high frequencies we need a gain of 10, so that's the ratio of the s^2 terms.
3. at low frequencies we need a gain of 1, so that's the ratio of the constant terms.
4. we need a guess at the dissipation, which is the reciprocal of the 'Q' of the denominator, and sets the denominator s^1 term. The filter covers a frequency range of about 1000:1. My guess is the *square root of this*, say 30.
5. at mid frequencies we need a gain of 0.4, and *very roughly indeed* we can set the ratio of the s^1 terms to this. In other words, I guess:

$$H(s) = \frac{10 \cdot s^2 + 0.4 * 30 \cdot s + 1}{1 \cdot s^2 + 30 \cdot s + 1} = \frac{10 \cdot s^2 + 12 \cdot s + 1}{1 \cdot s^2 + 30 \cdot s + 1} \qquad @\ 0.3Hz \qquad [3]$$

Check out the response – not bad, figure 4:

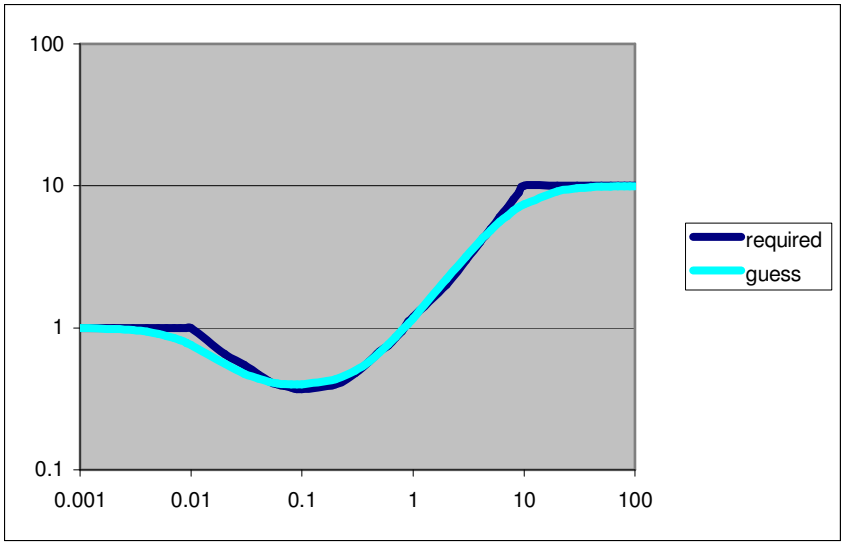


Figure 4: my initial guess

It's not perfect; let's get the spreadsheet to find a better fit. We need response limits, and a measure for how far out the result is. We'll use the "method of least squares". For any point whose gain goes outside set limits, take the square of the relative gain error (1-generated_gain/required_gain) at each frequency, and add 'em all up.

For the response limits shown in figure 5, I allowed 1% gain variation in the passband, and between 2xrequired and 0.1 in the bands outside. If a response point lies between these limits, it doesn't contribute to the error function.

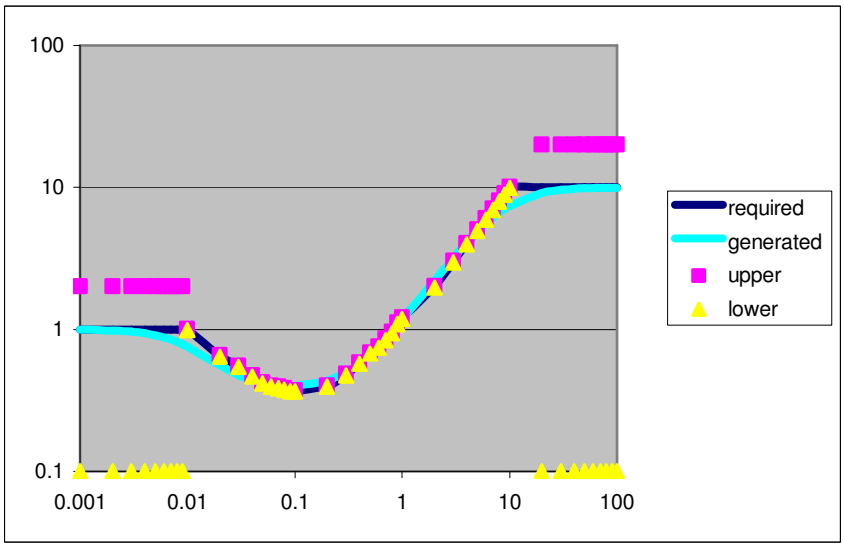


Figure 5: you've got to set limits

Now we use Excel's Solver to 'improve' our response. Configure it to minimize cell A2's value by adjusting the cells holding the filter coefficients (allowing only positive solutions), and click 'solve' to get figure 6:

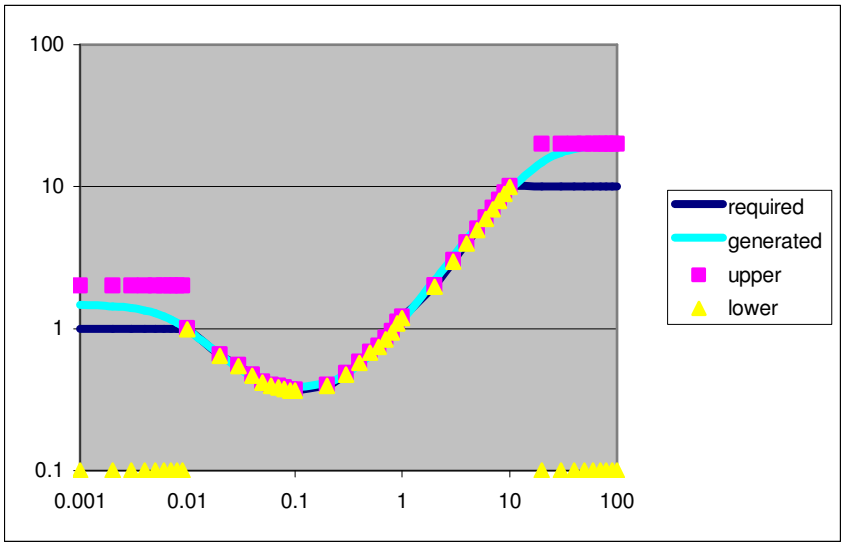


Figure 6: as if by magic, a better fit

Actually, I'm making it look a *little* easy. Excel's solver is prone to settling on a poor local minimum if your initial guess is too far out, so you sometimes need to intervene to help it.

This seems good in the desired frequency range, but that gain is still high outside the passband, especially at the higher frequencies. How about adding another filter section to chop off the low and high frequencies? Separate single pole lowpass and highpass filters would do. But did you realize that you can combine those single pole filters into one two-pole bandpass? It means we can reuse the formula we already created. I'll try a section with the same denominator as my first guess, but with only the s^1 term in the numerator non-zero, set to match the denominator:

$$H_2(s) = \frac{30 \cdot s}{1 \cdot s^2 + 30 \cdot s + 1} \qquad @\,0.3Hz \qquad [4]$$

Let's also be more aggressive with those upper gain limits, forcing them to fall outside the band, figure 7:

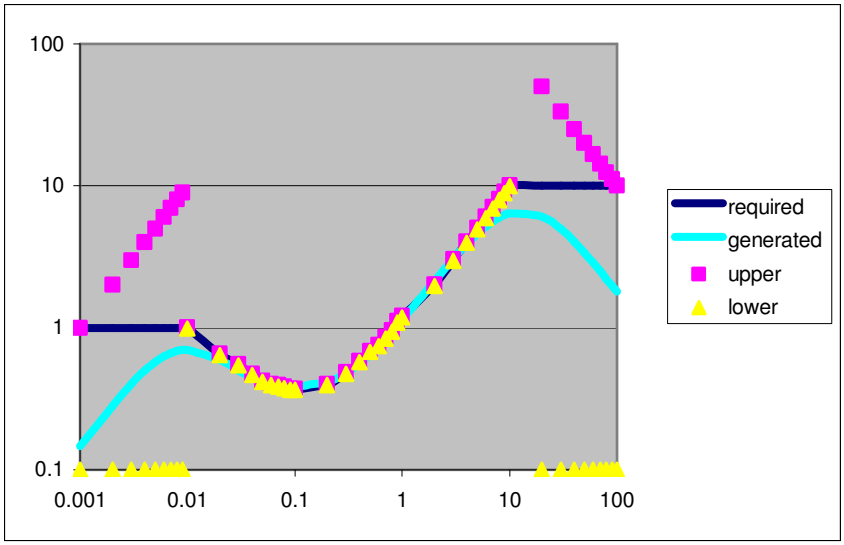
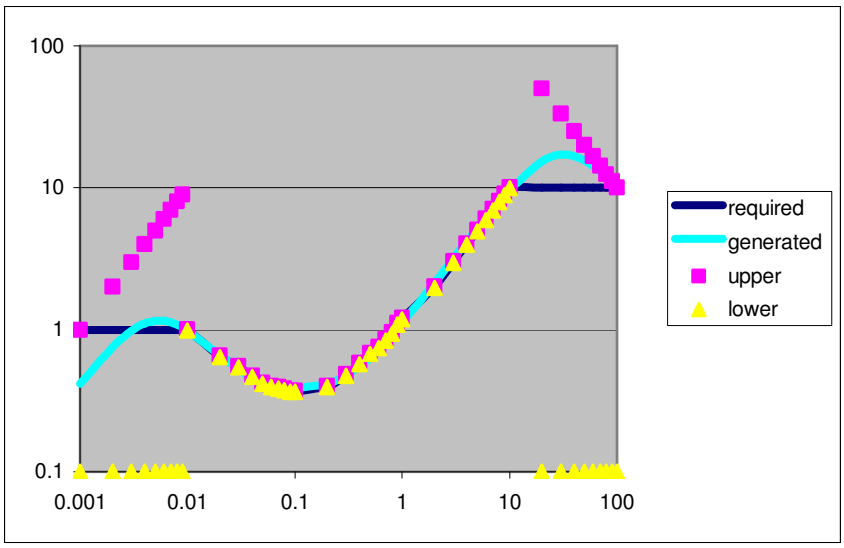


Figure 7: adding a bandpass to reduce the out-of-band stuff

We invoke the solver again, and get figure 8:



F*igure 8: final design with bandlimiting*

which is falls away nicely on both sides. It's still an analogue filter; in the next column we'll look at issues involved with turning it digital – in the same spreadsheet – then implementing it and analyzing it. Let me know if this technique could prove useful for *your* filter curve-fitting – Kendall.